LEARN C++
Skill up with our free tutorials

0.7 — Compiling your first program

👤 **ALEX**[1]   🕒 **NOVEMBER 28, 2023**

Before we can write our first program, we need to learn how to create new programs within our Integrated Development Environment (IDE). In this lesson, we'll cover how to do that, and you'll also compile and execute your first program!

Projects

To write a C++ program inside an IDE, we typically start by creating a new project (we'll show you how to do this in a bit). A **project** is a container that holds all of your source code files, images, data files, etc... that are needed to produce an executable (or library, website, etc...) that you can run or use. The project also saves various IDE, compiler, and linker settings, as well as remembering where you left off, so that when you reopen the project later, the state of the IDE can be restored to wherever you left off. When you choose to compile your program, all of the .cpp files in the project will get compiled and linked.

Each project corresponds to one program. When you're ready to create a second program, you'll either need to create a new project, or overwrite the code in an existing project (if you don't want to keep it). Project files are generally IDE specific, so a project created for one IDE will need to be recreated in a different IDE.
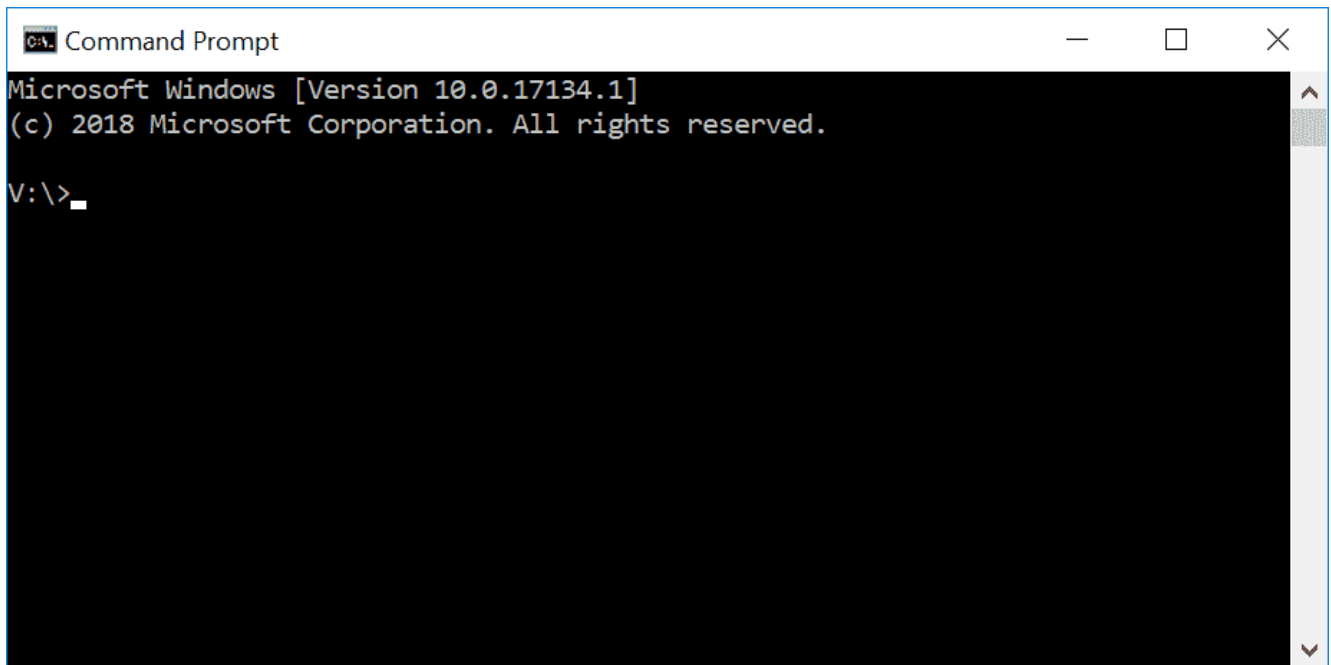
> **Best practice**
>
> Create a new project for each new program you write.

Console projects

When you create a new project, you'll generally be asked what type of project you want to create. All of the projects that we will create in this tutorial will be console projects. A **console project** means that we are going to create programs that can be run from the Windows, Linux, or Mac console.

Here's a screenshot of the Windows console:

By default, console applications have no graphical user interface (GUI), they print text to the console, read input from the keyboard, and are compiled into stand-alone executable files. This is perfect for learning C++, because it keeps the complexity to a minimum, and ensures things work on a wide variety of systems.

Don't worry if you've never used a console before, or don't know how to access it. We'll compile and launch our programs through our IDEs (which will invoke the console when necessary).

Workspaces / solutions

When you create a new project for your program, many IDEs will automatically add your project to a "workspace" or a "solution" (the term varies by IDE). A workspace or solution is a container that can hold one or more related projects. For example, if you were writing a game and wanted to have a separate executable for single player and multiplayer, you'd need to create two projects. It wouldn't make sense for both of these projects to be completely independent -- after all, they are part of the same game. Most likely, each would be configured as a separate project within a single workspace/solution.

Although you can add multiple projects to a single solution, we generally recommend creating a new workspace or solution for each program, especially while learning. It's simpler and there's less chance of something going wrong.

Writing your first program

Traditionally, the first program programmers write in a new language is the infamous [hello world program (https://en.wikipedia.org/wiki/Hello_world)][3], and we aren't going to deprive you of that experience! You'll thank us later. Maybe.

[()](#creating_project_vs19)Creating a project in Visual Studio 2019 (or newer) 🔗 [4]
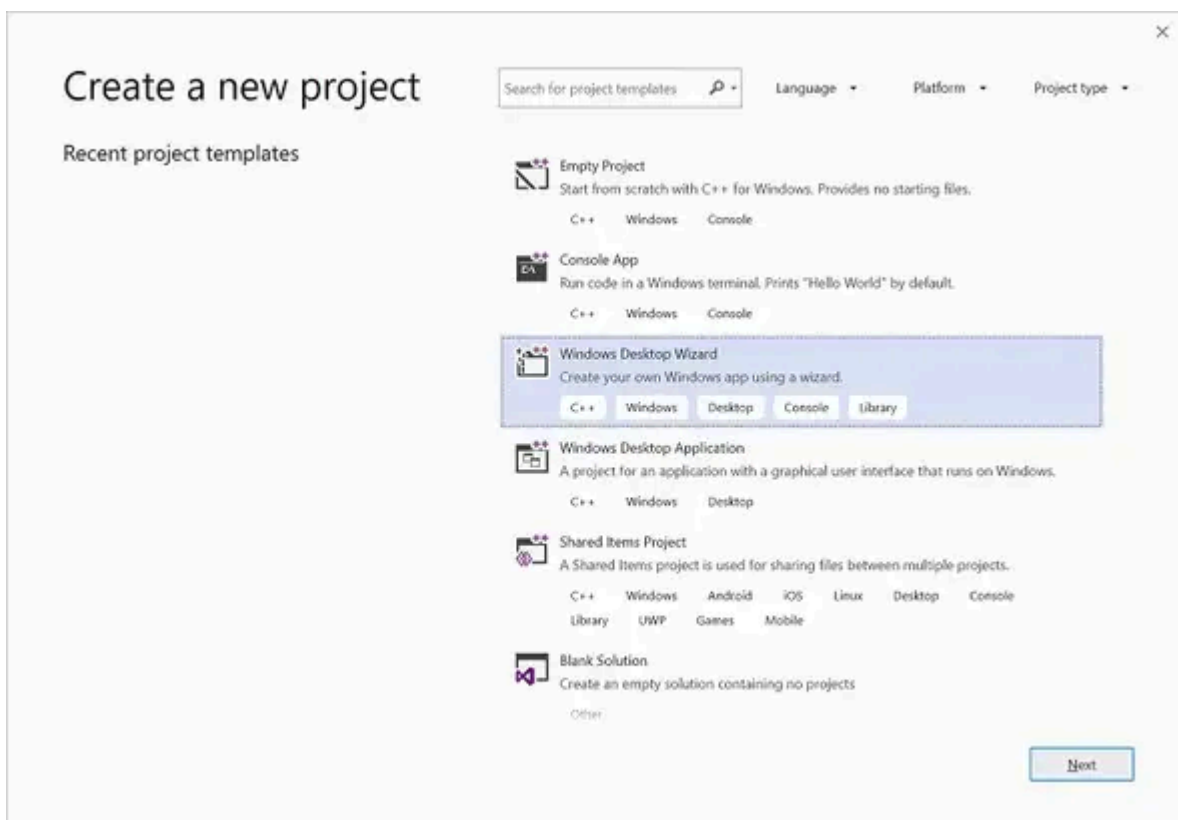
When you run Visual Studio 2019 (or newer), you should see a dialog that looks like this:
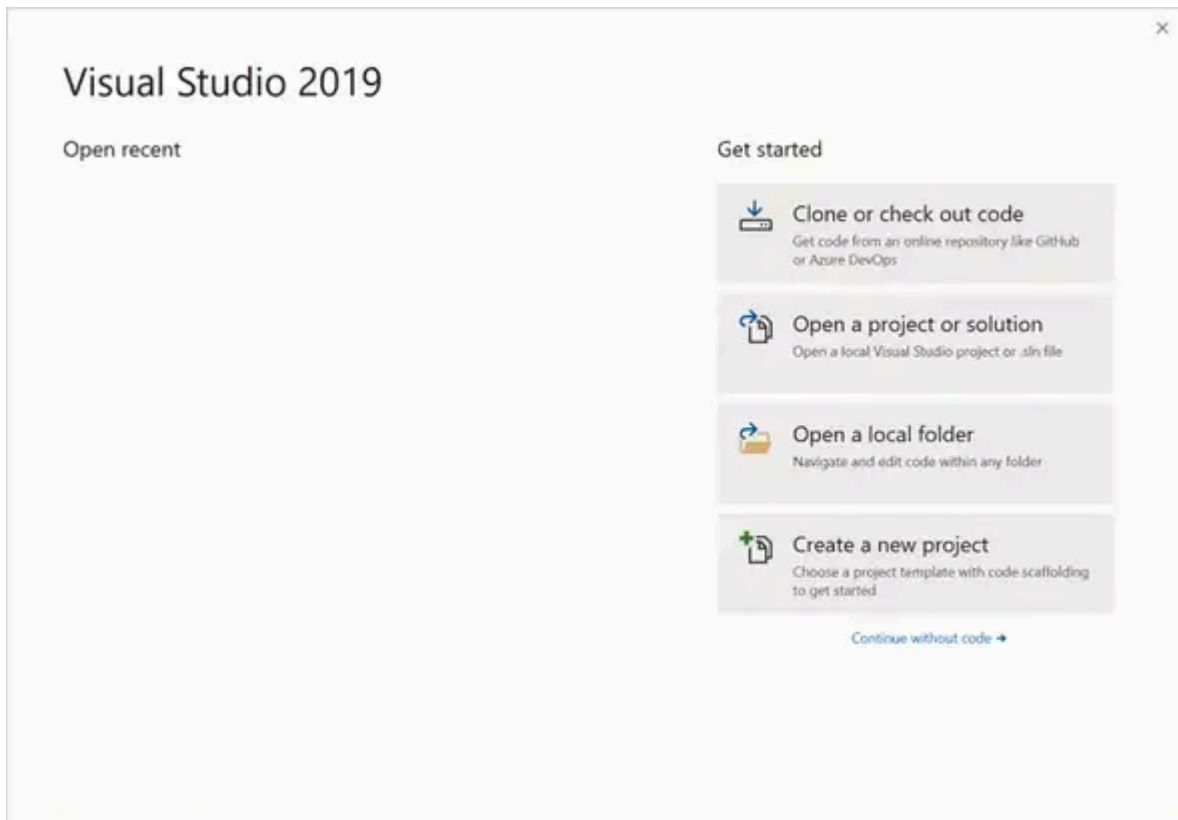
Select *Create a new project*.

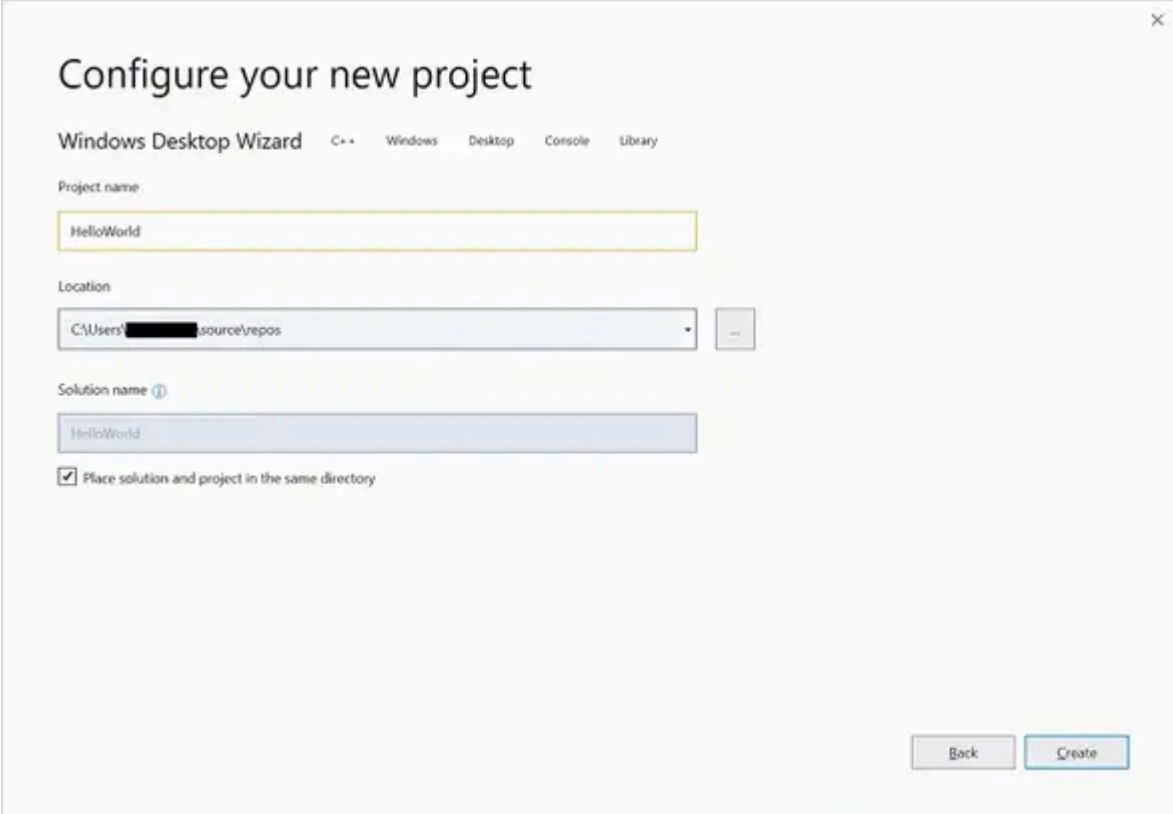You'll then see a dialog that looks like this:

If you've already opened a prior project, you can access this dialog via the *File menu > New > Project*.

Select *Windows Desktop Wizard* and click *Next*. If you don't see this, then you probably forgot to choose to install the *Desktop development with C++* workload when you installed Visual Studio. In that case, go back to lesson 0.6 -- Installing an Integrated Development Environment (IDE)

(https://www.learncpp.com/cpp-tutorial/installing-an-integrated-development-environment-ide/)[5] and reinstall your Visual Studio as indicated (note: rather than doing a full reinstall, you can run the Visual Studio installer and modify your existing installation to add the C++ workload).

Next, you'll see a dialog that looks like this:

Replace the existing project name with `HelloWorld`.

It's recommended that you also check the *Place solution and project in the same directory*, as this reduces the number of subdirectories that get created with each project.

Click *Create* to continue.

Finally, you'll see one last dialog:
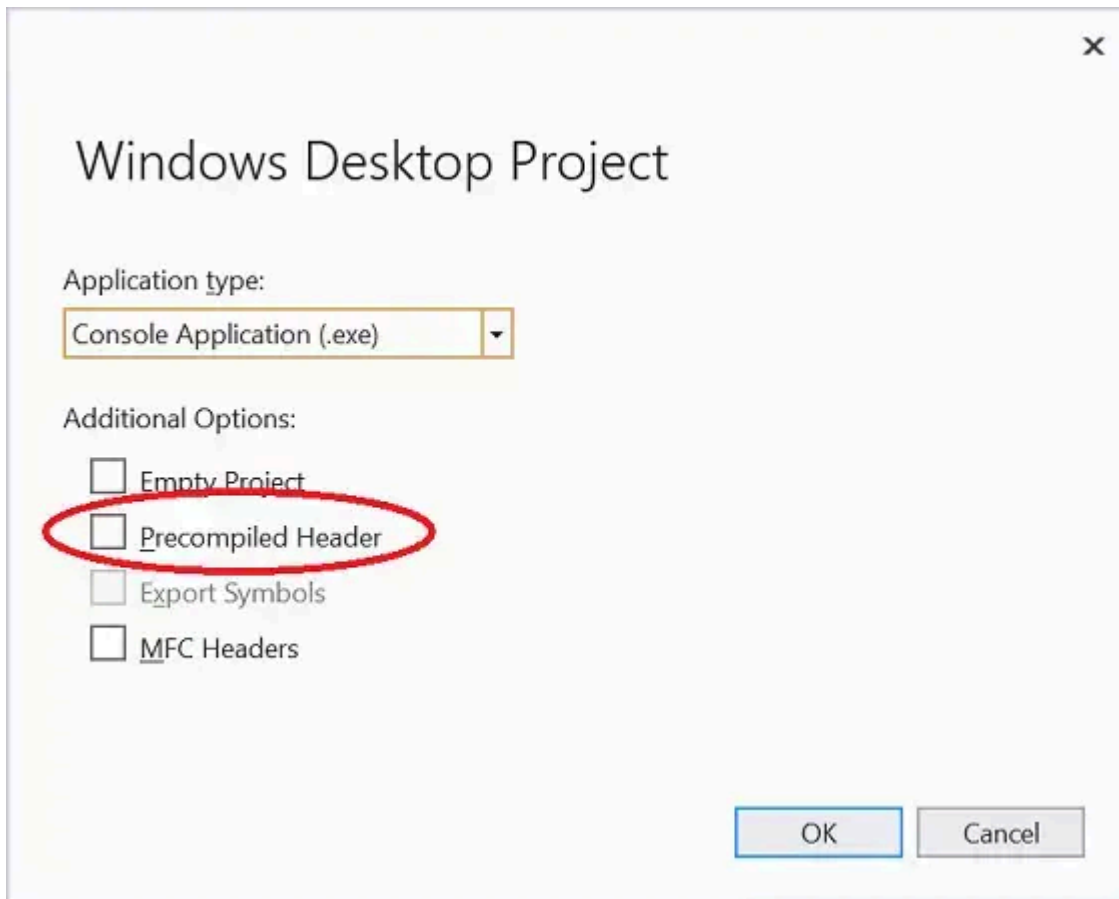
Make sure the *Application type* is set as *Console Application (.exe)* and that the *Precompiled Header* option is unselected. Then click *OK*.

You've now created a project! Jump down to the Visual Studio Solution Explorer (#visual_studio_solution_explorer)[6] section below to continue.

> ## Q: What are precompiled headers and why are we turning them off?
>
> In large projects (those with many code files), precompiled headers can improve compilation speed by avoiding some redundant compilation that tends to occur in larger projects.
>
> However, precompiled headers require extra work to use, and for small projects (such as those you'll create in our tutorials) make little to no difference in compilation times.
>
> For this reason, we recommend turning precompiled headers off initially, and only enabling them later if and when you find your compilation times suffering.

Creating a project in Visual Studio 2017 or older

To create a new project in Visual Studio 2017 or older, go to the *File menu > New > Project*. A dialog box will pop up that looks something like this:
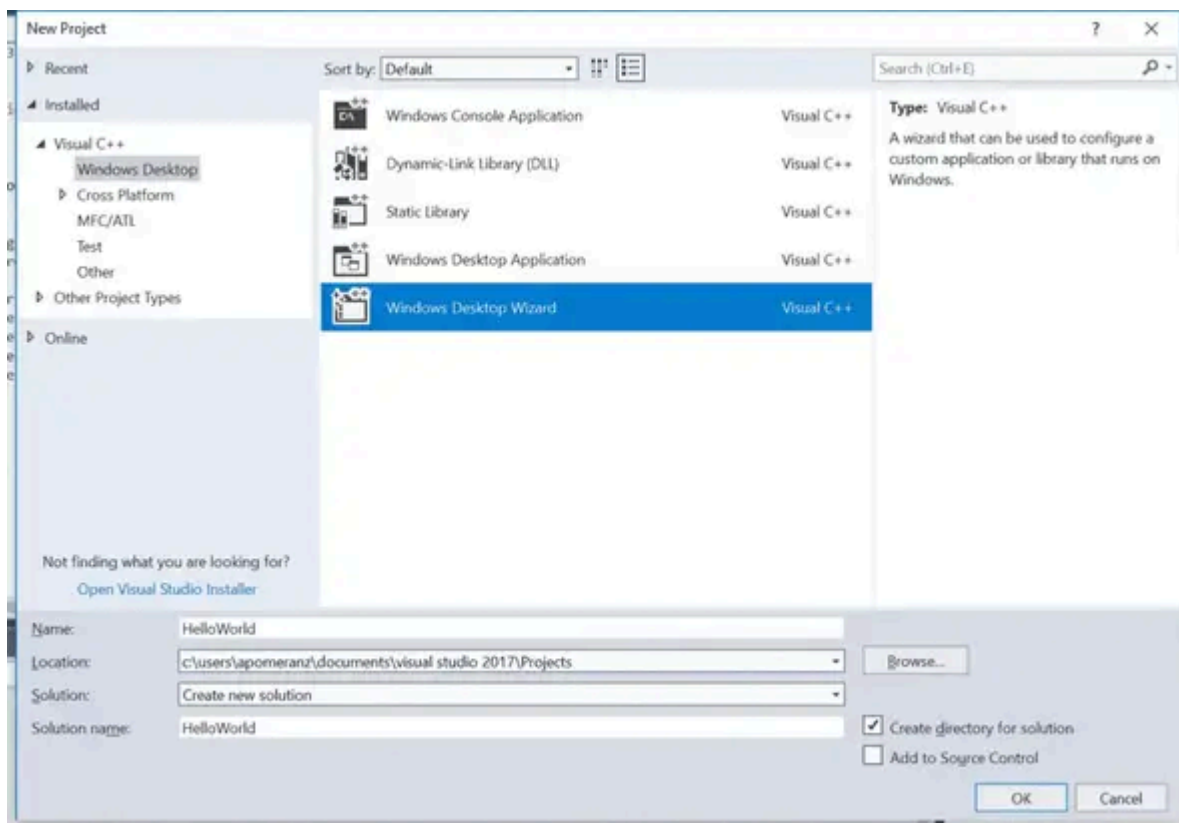
First, make sure *Visual C++* is listed on the left side. If you don't see *Visual C++*, then you probably forgot to choose to install the *Desktop development with C++* workload when you installed Visual Studio. In that case, go back to lesson 0.6 -- Installing an Integrated Development Environment (IDE) (https://www.learncpp.com/cpp-tutorial/installing-an-integrated-development-environment-ide/)[5] and reinstall your Visual Studio as indicated (note: rather doing a full reinstall, you can run the Visual Studio installer and modify your existing install to add the C++ workload).

If you're using Visual Studio 2017 v15.3 or newer, underneath *Visual C++*, select *Windows Desktop* and then select *Windows Desktop Wizard* in the main window.

If you don't see *Windows Desktop* as an option, you're probably using an older version of Visual Studio. That's fine. Instead, choose *Win32* and then *Win32 Console Application* in the main window.

Down below, in the *Name* field, enter the name of your program (replace the existing name with `HelloWorld`). In the Location field, you can optionally select a different location for your project to be placed into. The default is fine for now.

Click *OK*. If you're using an older version of Visual Studio, the Win32 Application Wizard will launch. Press *Next*.

At this point, you should see a wizard dialog that looks something like this (older versions of Visual Studio use a different style, but have most of the same options):
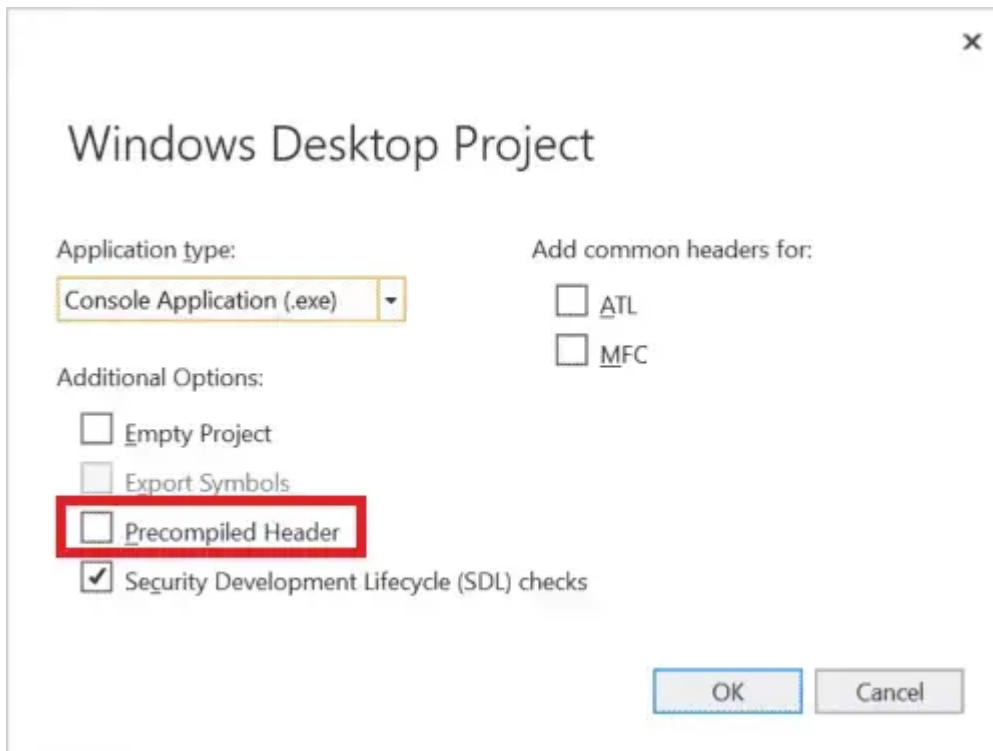
Make sure you uncheck *Precompiled Header*.

Then click *Ok* or *Finish*. Now your project is created!

()Visual Studio Solution Explorer 🔗 (#visual_studio_solution_explorer)[6]

On the left or right side of the window, you should see a window titled *Solution Explorer*. Inside this window, Visual Studio has created a solution for you (*Solution 'HelloWorld'*). Within that, with the name in bold, is your new project (*HelloWorld*). Within the project, Visual Studio has created a number of files for you, including *HelloWorld.cpp* (underneath the *Source Files* tree item). You may also see some other .cpp or .h files, which you can ignore for now.
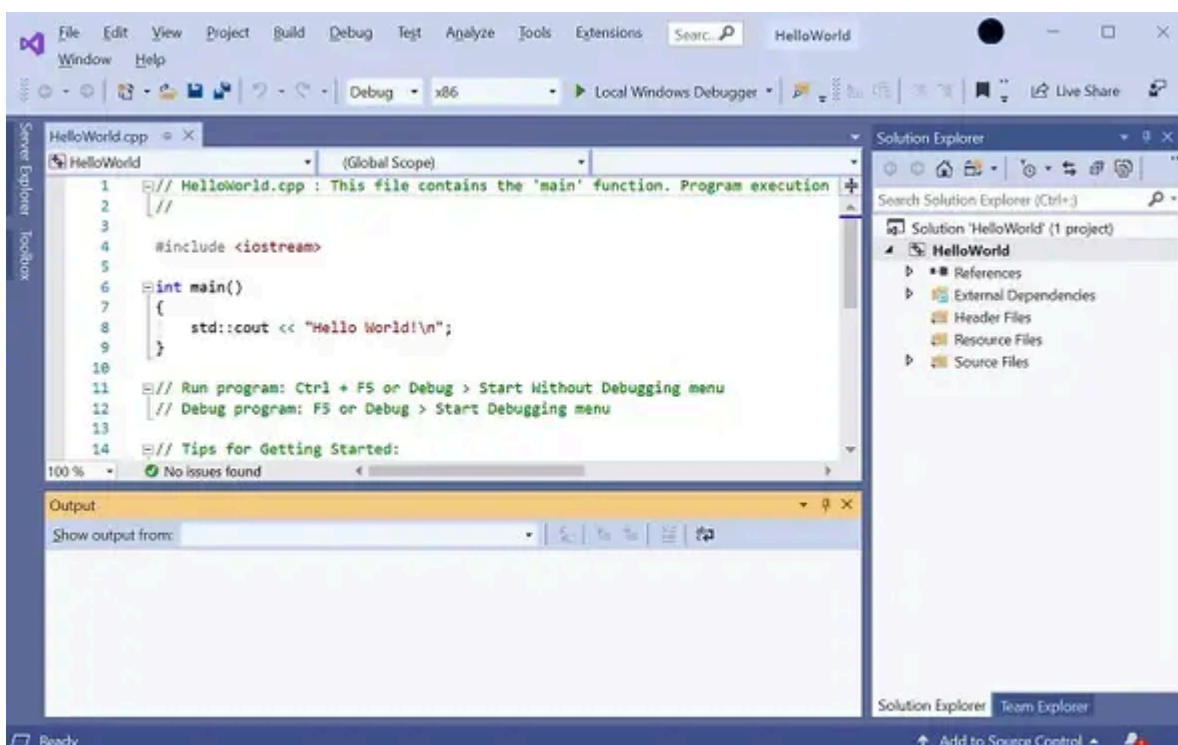
In the text editor, you will see that Visual Studio has already opened *HelloWorld.cpp* and created some code for you. Select and delete all of the code, and type/copy the following into your IDE:

```
1   #include <iostream>
2
3   int main()
4   {
5       std::cout << "Hello, world!";
6       return 0;
7   }
```

To compile your program, either press *F7* (if this doesn't work, try *Ctrl-Shift-B*) or go to the *Build menu > Build Solution*. If all goes well, you should see the following appear in the Output window:

```
1>------ Build started: Project: HelloWorld, Configuration: Debug Win32 --
1>HelloWorld.cpp
1>HelloWorld.vcxproj → c:\users\alex\documents\visual studio 2017\Projec
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ========
```

This means your compile was successful!

> **Q: I got error C1010 ("fatal error C1010: unexpected end of file while looking for precompiled header. Did you forget to add '#include "stdafx.h"' to your source?"). What now?**
>
> You forgot to turn off precompiled headers when you created your project. Recreate your project (as per the [instructions above (#creating_project_vs19)](#creating_project_vs19)[4]) and make sure to disable precompiled headers.

To run your compiled program, press *Ctrl-F5*, or go to the *Debug menu* and choose *Start Without Debugging*. You will see something like the following:

That is the result of your program! Congratulations, you've compiled and run your first program!

Related content

When you run a program directly from Visual Studio, you may see an additional line of output that looks something like this:

```
C:\Users\Alex\source\repos\Project6\Debug\Project6.exe (process 21896)
◀                                                                                    ▶
```

This is normal. Visual Studio is providing some additional information about whether your program exited normally or abnormally. We discuss this further in lesson 2.2 -- Function return values (value-returning functions) (https://www.learncpp.com/cpp-tutorial/function-return-values-value-returning-functions/)[7].

Creating a project in Code::Blocks

To create a new project, go to *File menu > New > Project*. A dialog box will pop up that looks like this:
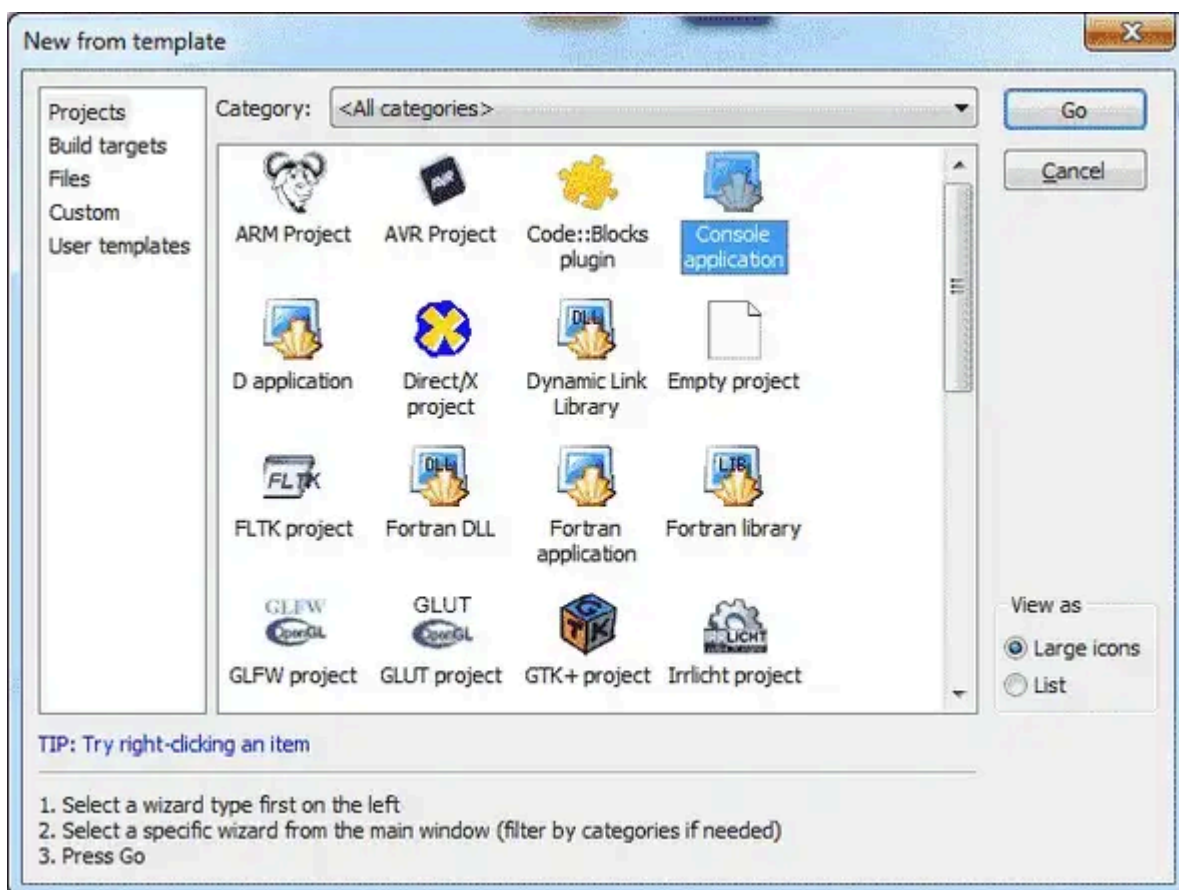
Select *Console application* and press the *Go (or Create)* button.

If you see a console application wizard dialog, press *Next*, make sure C++ is selected and press *Next* again.

Now you will be asked to name your project. Title the project `HelloWorld`. You can save it wherever you wish. On Windows, we recommend you to save it in a subdirectory of the C drive, such as `C:\CBProjects`.
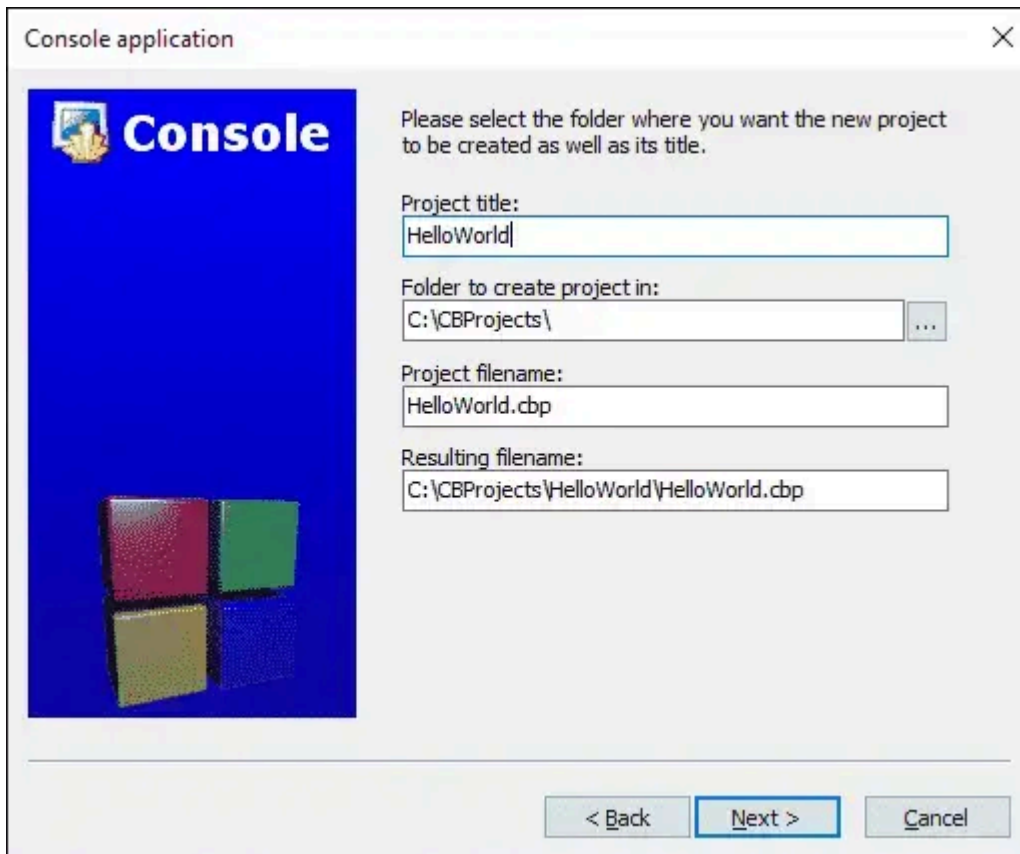
You may see another dialog asking you which configurations you want enabled. The defaults should be fine here, so select *Finish*.

Now your new project has been created.

On the left side of the screen, you should see a *Management* window, with the *Projects* tab selected. Inside that window, you'll see a *Workspace* folder, with your *HelloWorld* project inside of it:
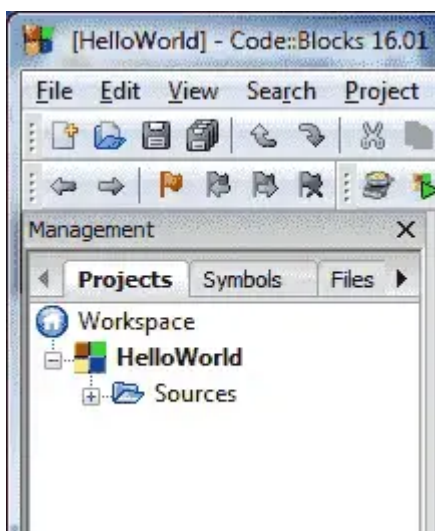
Inside the *HelloWorld* project, expand the *Sources* folder, and double click on "main.cpp". You will see that a hello world program has already been written for you!

Replace that one with the following:

```
1  #include <iostream>
2
3  int main()
4  {
5      std::cout << "Hello, world!";
6      return 0;
7  }
```

To build your project, press *Ctrl-F9*, or go to *Build menu > Build*. If all goes well, you should see the following appear in the Build log window:

```
-------------- Build: Debug in HelloWorld (compiler: GNU GCC Compiler)---
mingw32-g++.exe -Wall -fexceptions -g -std=c++14  -c C:\CBProjects\HelloW
mingw32-g++.exe  -o bin\Debug\HelloWorld.exe obj\Debug\main.o
Output file is bin\Debug\HelloWorld.exe with size 1.51 MB
Process terminated with status 0 (0 minute(s), 0 second(s))
0 error(s), 0 warning(s) (0 minute(s), 0 second(s))
```

This means your compile was successful!

To run your compiled program, press *Ctrl-F10*, or go to *Build menu > Run*. You will see something similar to the following:
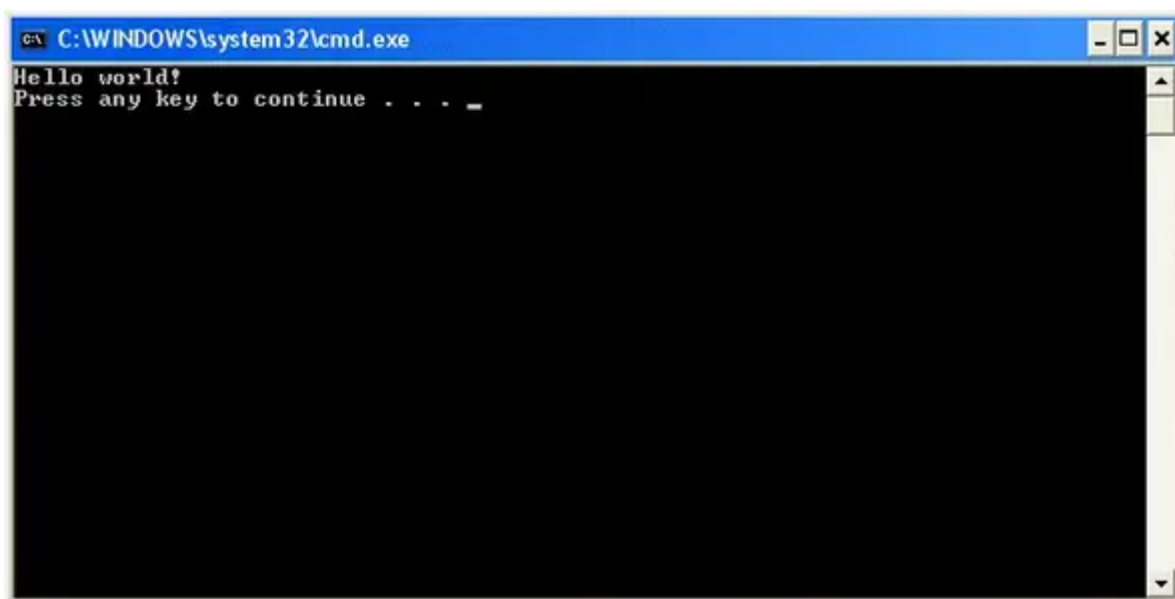
That is the result of your program!

For Linux users

Linux users may need to install the additional packages before Code::Blocks will compile. Please see the Code::Blocks installation instructions in lesson [0.6 -- Installing an Integrated Development Environment (IDE) (https://www.learncpp.com/cpp-tutorial/installing-an-integrated-development-environment-ide/)][5] for more info.

Creating a project in VS Code

To start a new project, go to the *View > Explorer* menu (or press *Ctrl-Shift-E*). This will open the explorer pane. If you haven't previously opened a project, you should see an *Open Folder* button in the explorer pane -- press it. If there is already an open project and you want to start a new one, choose *File > Open Folder* from the top nav.

Inside the dialog that opens, create a new folder named *HelloWorld* and then select this folder. This folder will be your project folder.

Next, we need to create the file that will contain our source code. Choose *File > New File* from the top nav, or click the *New File icon* to the right of *HELLOWORLD* in the explorer pane.

Name your file *main.cpp* and add the following contents to it:

```
1   #include <iostream>
2
3   int main()
4   {
5       std::cout << "Hello, world!";
6       return 0;
7   }
```

To compile *main.cpp* and run the program, make sure *main.cpp* is open in the main pane, and then either choose *Run > Run Without Debugging* from the top nav, or click the *v* to the right of the play icon to the right of *main.cpp* tab and choose *Run C/C++ File*.

Next, choose the *g++ build and debug active file* option (macOS users should choose *clang++* instead of *g++*). Switch the tab from *DEBUG CONSOLE* to *TERMINAL* at the bottom of the window.

If the terminal contains the text "Hello, world!", then congratulations, you've just run your first C++ program!

If you're using g++ on the command line

In this case, you don't need to create a project. Simply paste the following into a text file named HelloWorld.cpp and save your file:

```
1   #include <iostream>
2
3   int main()
4   {
5       std::cout << "Hello, world!";
6       return 0;
7   }
```

From the command line, type:

```
g++ -o HelloWorld HelloWorld.cpp
```

This will compile and link HelloWorld.cpp. To run it, type:

`HelloWorld` (or possibly `./HelloWorld`), and you will see the output of your program.

If you're using other IDEs or a web-based compiler

You will have to figure out how to do the following on your own:

1. Create a console project (IDEs only)
2. Add a .cpp file to the project (IDEs only, if one isn't auto-created for you)
3. Paste the following code into the file:

```
1   #include <iostream>
2
3   int main()
4   {
5       std::cout << "Hello, world!";
6       return 0;
7   }
```

4. Compile the project
5. Run the project

If compiling fails

It's okay, take a deep breath. We can probably fix it. :)

First, look at the error message that the compiler gave you. Most often, it will contain a line number indicating which line was in error. Examine both that line and the lines around it, and make sure there are no typos or misspellings. Also make sure you're not including line numbers in your code (your editor should be providing those).

Second, look at the Q&A in lesson 0.8 -- A few common C++ problems (https://www.learncpp.com/cpp-tutorial/a-few-common-cpp-problems/)[8], as your issue may be covered there.

Third, read the comments below the lesson containing the example you're compiling -- someone may have encountered the same issue and provided a solution.

Finally, if all of the above fail, try searching for your error message on your favorite search engine. It's likely someone else has encountered this issue before and figured out how to fix it.

If your program runs but the console window flashes and closes immediately

When a console program is run, the console window will open and any output from the program will be written into the console window.

When the program has finished running, most modern IDEs will keep the console open (until you press a key) so you can inspect the results of the program before continuing. However, some older IDEs will automatically close the console window when the program finishes running. This is generally not what you want.

If your IDE closes the console window automatically, the following two steps can be used to ensure the console pauses at end of the program.

First, add or ensure the following lines are near the top of your program:

```
1  #include <iostream>
2  #include <limits>
```

Second, add the following code at the end of the main() function (just before the return statement):

```
1  std::cin.clear(); // reset any error flags
2  std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); //
   ignore any characters in the input buffer until we find a newline
   std::cin.get(); // get one more char from the user (waits for user to
3  press enter)
```

This will cause your program to wait for the user to press enter before continuing (you may have to press enter twice), which will give you time to examine your program's output before your IDE closes the console window.

Other solutions, such as the commonly suggested `system("pause")` solution may only work on certain operating systems and should be avoided.

If the console window doesn't open at all and your program doesn't appear to be running, your anti-virus or anti-malware may also be blocking execution of the program. If that's the case, try temporarily disabling your scanners and see if the problem resolves, or exclude the project directory from your scans.

> **Tip**
>
> If you're using anti-virus or anti-malware, consider excluding your project directory from the scans, as compiling and debugging may cause false positives.

What is the difference between the compile, build, rebuild, clean, and run/start options in my IDE?

In lesson 0.5 -- Introduction to the compiler, linker, and libraries (https://www.learncpp.com/cpp-tutorial/introduction-to-the-compiler-linker-and-libraries/)[9], we showed that to produce an executable that can be run, each code file in a program is compiled into an object file, and then the object files are linked into an executable.

When a code file is compiled, your IDE may cache the resulting object file. That way, if the program is compiled again in the future, any code file that hasn't been modified doesn't need to be recompiled -- the cached object file from last time can be reused. This can speed up compilation times significantly (at the cost of a little bit of disk space).

With that in mind, here's what each of the options typically does:

- **Build** compiles all *modified* code files in the project or workspace/solution, and then links the object files into an executable. If no code files have been modified since the last build, this option does nothing.
- **Clean** removes all cached objects and executables so the next time the project is built, all files will be recompiled and a new executable produced.
- **Rebuild** does a "clean", followed by a "build".
- **Compile** recompiles a single code file (regardless of whether it has been cached previously). This option does not invoke the linker or produce an executable.
- **Run/start** executes the executable from a prior build. Some IDEs (e.g. Visual Studio) will invoke a "build" before doing a "run" to ensure you are running the latest version of your code. Otherwise (e.g. Code::Blocks) will just execute the prior executable.

Although we talk informally about "compiling" our programs, to actually compile our programs we will typically choose the "build" (or "run") option in our IDE to do so.

Conclusion

Congratulations, you made it through the hardest part of this tutorial (installing the IDE and compiling your first program)!

Don't worry if you don't understand what the different lines in the Hello World program do. We'll examine and explain each line in detail at the start of the next chapter.

Next lesson
0.8   A few common C++ problems

8

Back to table of contents

10

Previous lesson
0.6   Installing an Integrated Development Environment (IDE)

5

11

B   U   URL   INLINE CODE   C++ CODE BLOCK   HELP!

Leave a comment...

👤 Name*

@ Email*   ⑦

🐞 Find a mistake? Leave a comment above!⑦

👤 Avatars from https://gravatar.com/[13] are connected to your provided email address.

Notify me about replies: 🔔

POST COMMENT

1.1K COMMENTS

Newest ▾

Tomás Bernardo
🕐 April 4, 2024 4:59 am

I think there's a typo here (like the absence of a letter) "pane"
https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#:~:text=open%20the%20explorer-,pane,-.%20If%20you%20haven%E2%80%99t

👍 0   ↪ Reply

> **Alex**   Author
> 💬 Reply to Tomás Bernardo [14]   🕐 April 4, 2024 2:09 pm
>
> Microsoft calls it a pane: https://learn.microsoft.com/en-us/visualstudio/ide/use-solution-explorer?view=vs-2022
>
> Panel would probably work too.
>
> 👍 0   ↪ Reply

Muhammad Essameldin
🕐 March 29, 2024 5:54 pm

thanks

👍 0   ↪ Reply

PaxSectoid

○ March 12, 2024 3:39 am

I'm using VSCode on windows and the output doesn't display on the terminal.

👍 0   ↪ Reply

jazzymaxine   🔗

💬 Reply to PaxSectoid [15]   ○ March 24, 2024 3:25 am

https://www.freecodecamp.org/news/how-to-write-and-run-c-cpp-code-on-visual-studio-code/

tl;dr install code runner extension, then in settings search "run code in terminal" and activate the setting "Code-runner: Run In Terminal". Then run your cpp file with the "run code" option, *not* the "run c/c++ file" option.

👍 0   ↪ Reply

Carlos Manuel Martínez Zumaya
○ March 9, 2024 12:21 pm

Salutations. Sorry for being a bother yet again.

I wanted to ask, since we are at the compiling topic:

Is the famous **Dragon Book**, also known as **Compilers: Principles, Techniques, and Tools** still a reliable source to learn about how to write a compiler?

After finishing this tutorial, I would like my first big project to be writing a compiler; I am even learning some Assembly along C++ and parsing theory. I have heard this book in particular is the absolute Bible about how compilers works and are written. But I can appreciate the most "recent" publication is from the year 2006, although I may be misunderstanding something about this information. The copies of this work are pretty expensive at my country (even used ones are pretty costly) yet I do not want to get a pirated version or something like that; I am from Mexico, if the input is needed. So I would prefer to be sure if the content of the text is still updated.

Of course, I do not plan of doing it because I want it to be the next GCC, Clang or something like that. It is just for pedagogic/enjoyment matters.

Thank you again for this sublime learning resource; you are an awesome individual.

👍 0   ↪ Reply
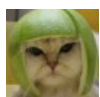
Alex   Author

💬 Reply to Carlos Manuel Martínez Zumaya [16]   ○ March 11, 2024 5:32 pm

I have no idea. I haven't read it, and writing compilers isn't really my thing. Sorry. :)

👍 1   ↪ Reply

Carlos Manuel Martínez Zumaya

💬 Reply to Alex [17]   🕐 March 12, 2024 9:17 pm

It is ok, I understand Mr. Alex. I still thank you very much for taking the time to answer.

I am asking for information about the text to friends with a Computer Science degree, but they neither have read the book. I guess I will have to do more research.

Thank you so much again for lending your time to the community!

👍 1   ↪ Reply

Bryant
🕐 February 27, 2024 2:55 am

```
g++ HelloWorld.cpp -o HelloWorld
```
i think this is the correct syntax

👍 0   ↪ Reply

JPerk
🕐 February 25, 2024 6:37 am

Had a bit of a tough time following along with vim and the terminal but it was fun.

👍 0   ↪ Reply

Dejan Djorovic
🕐 February 23, 2024 6:55 pm

While I build first code on VISUAL STUDIO 2022, Hello World, after build solution code was not do his job and VS ask me : " did you forget to add ´#include "pch.h".And also something about precompiled header file.Can anyone help me, what shod Ido?

👍 0   ↪ Reply

Scout tf2

💬 Reply to Dejan Djorovic [18]   🕐 February 24, 2024 12:47 pm

It's a small mistake in the guide, create another project, then disable precompiled headers, they're fairly useless for now

👍 1   ↪ Reply

Preston Love
🕐 February 16, 2024 8:38 am

I'm using VScode on a mac and keep getting this error when running: "linker command failed with exit code 1 (use -v to see invocation)"

I have no idea what the hell I'm doing.

👍 0   ↪ Reply

> **jazzymaxine**
> 💬 Reply to Preston Love [19]   🕐 March 24, 2024 3:29 am
>
> did you set your compiler to g++? that's what I'm using. Check that you have clang installed, too, from earlier in this tutorial.
>
> 👍 0   ↪ Reply

> **jazzymaxine**
> 💬 Reply to Preston Love [19]   🕐 March 24, 2024 3:28 am
>
> that's interesting, it works for me on MacOS Monterey. Try following this tutorial, it's for windows but it worked for me. Did you also install code runner and c++ extension pack? I did not get your error though so this might be unheplful information. https://www.freecodecamp.org/news/how-to-write-and-run-c-cpp-code-on-visual-studio-code/
>
> 👍 0   ↪ Reply

> **Deepu**
> 💬 Reply to Preston Love [19]   🕐 March 4, 2024 8:03 am
>
> In previous chapter they said vs on mac doesn't support c++ may be that's why.
>
> 👍 0   ↪ Reply

Gabriel
🕐 February 14, 2024 5:56 am

I'm using cxxdroid for mobile. Should I pay attention to something so that my project can be compiled in other environments, such as Visual Studio?

✏️ *Last edited 1 month ago by Gabriel*

👍 0   ↪ Reply

Daur Daur Toogis

🕐 February 10, 2024 1:07 pm

I'm using VScode on mac. Do I need to install a compiler like Clang as well?

👍 0   ↪ Reply

> **Alex**   Author
>
> 💬 Reply to Daur Daur Toogis [20]   🕐 February 11, 2024 11:07 pm
>
> See https://code.visualstudio.com/docs/cpp/config-clang-mac
>
> 👍 1   ↪ Reply

RESS

🕐 February 7, 2024 4:14 pm

Love ya alex, making my cyber and network security classes at school look useless.

👍 1   ↪ Reply

IceFloe

🕐 February 4, 2024 2:22 pm

Thanks for the lesson, everything seems to be working without errors, and I more or less understand the process of how the IDE works, as I understand it, an Internet connection is not necessary, all the necessary parameters and libraries are built into the IDE itself? And I also wanted to clarify, will we be working with console applications throughout the course, or are there other options for launching and working on the written code?

👍 0   ↪ Reply

> **Alex**   Author
>
> 💬 Reply to IceFloe [21]   🕐 February 5, 2024 3:49 pm
>
> Yes, IDEs are typically self-contained, meaning once they are installed you will not need internet access.
>
> You can build other types of applications besides console applications, but doing so is beyond the scope of these tutorials.
>
> 👍 1   ↪ Reply

TahVonun

🕐 January 31, 2024 10:01 pm

Hello !

So I would like to start coding using Kali Linux and I know there is Vim used to code and g++ used as a compiler . My issue is that I dont know how to save the code i am writting using Vim . If anyone have an answer ? And maybe I should just follow the exact steps here and relearn later how to use Kali Linux and its tools ?

👍 0   ↪ Reply

Patrik V.

💬 Reply to TahVonun [22] 🕐 February 3, 2024 4:02 am

Use :w (write) to save the code/text with VIM.

👍 1   ↪ Reply

Kamil G.

🕐 January 27, 2024 10:31 am

(° ⌡°)

✏️ *Last edited 2 months ago by Kamil G.*

👍 1   ↪ Reply

S.E.

🕐 January 23, 2024 11:26 pm

You mention in this lesson that you will only be covering console-based programs. So I assume that means this site does not, and will not cover and form of GUI or drawing to the screen?

If so, will you later refer to good resources for learning those things too?

I am really interested in doing things like replicating falling-sand games and life simulations (like conways). I have actually made a cellular automaton in Godot and GDScript, but that comes with many built in tools and features etc, so it feels like cheating.

I really want to be able to hand craft small little demos for my own learning and fun, but I was never really sure where to begin. That is why I'm starting here with C++.

👍 1   ↪ Reply

Carlos Manuel Martínez Zumaya

💬 Reply to S.E. [23] 🕐 March 9, 2024 2:43 pm

Salutations, S.E.
I hope you are having an awesome day.

Maybe, these references will help you in your journey:

https://www.cprogramming.com/game-programming.html?inl=nv
https://www.cprogramming.com/graphics-programming.html?inl=nv

I will not say they are the absolutely best of the best, but they are truly useful.

Also, it is not a bad idea checking the documentation of the tools you will choose for your projects, e.g. the **OpenGL** specification:

https://www.khronos.org/opengl/wiki/

That Wiki page has a lot of useful resources!

If creating projects similar of Conway's Game of Life is your goal, learning **Automata Theory** is a must. Here are some interesting lectures about this topic:

https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/automata-theory/basics.html
https://cs.lmu.edu/~ray/notes/automatatheory/

It is a branch of Mathematics and, not gonna lie to you, **it can become really tricky to learn and master**; but the journey is, indeed, marvelous.

Also, once you feel confident enough with your programming skills, a great way of learning is **actually exploring the source code of Open Source (excuse redundancy) projects and studying how the code works**:

https://github.com/SuperTux/supertux

Completely made with C++! If you are in the mood, you can even contribute to this and more projects. This will increment your experience and you can even add your contributions to your programmer portfolio.

Remember that learning is not just about following the tutorial; **Is about initiative, curiosity and researching lots and lots.**

I hope I helped you with your mission. Remember: **Do not give up. Your effort will bring you awesome things!**

P.D. Sorry for being late...

👍 0   ↪ Reply
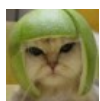
Alex   Author
💬 Reply to S.E. 23   🕐 January 25, 2024 3:09 pm

> So I assume that means this site does not, and will not cover and form of GUI or drawing to the screen?

Correct. The C++ standard library doesn't provide facilities for those things.

> If so, will you later refer to good resources for learning those things too?

I mention some things in https://www.learncpp.com/cpp-tutorial/appendix-c-the-end/ but definitely not exhaustively. The best library is really use-dependent.

🖒 2   ↪ Reply

Saphire

💬 Reply to Alex [24]   ⓘ January 30, 2024 5:43 pm

Thank god i've read this comment today, i'll take a look at the link you sent there, i'm also intrusted in creating GUIs, interfaces etc, with C++ for games and stuff. I've been learning with your website since months ago (actually, i think a year ago), but i've give up, now i'm restarting everything again, and trusting your website! Because i don't remember much things and coding at all, my plans are to create hacks for games, thanks for all those tutorials.

🖒 1   ↪ Reply

Dwina Daugherty
ⓘ January 16, 2024 4:55 am

Everything fine on both my Windows 11 laptop and my Windows 10 desktop.

🖒 1   ↪ Reply

"SuperPro"
ⓘ January 9, 2024 11:52 am

When I was building the program, it would give me a warning, stating `error C2855: command-line option '/std:c++' inconsistent with precompiled header` If anyone knows how to fix the problem, please tell me as I am unable to compile new builds. Thanks!

✎ *Last edited 2 months ago by "SuperPro"*

🖒 0   ↪ Reply

Murmur
ⓘ January 8, 2024 12:40 pm

The door is open

🖒 1   ↪ Reply

Justin

🕐 January 7, 2024 8:22 am

Im using Clion on mac and when I click the run it builds and runs the program together on the bottom. Am i supposed to run the program in the terminal and if so how. Thanks

👍 1   ↪ Reply

> **Peter**
> 💬 Reply to Justin [25]   🕐 January 12, 2024 8:56 am
>
> You good brother, clion has a built in terminal, which is at the bottom.
>
> 👍 0   ↪ Reply

> **Harriz**
> 💬 Reply to Justin [25]   🕐 January 10, 2024 11:02 pm
>
> Same for me but on Windows
>
> 👍 1   ↪ Reply

Ground Control
🕐 January 1, 2024 6:24 am

a little annoying how it says continue with a link to 2.2. Someone who is following along might accidently think that's the next chapter, read a few instructions, and have no idea what anything is.

👍 0   ↪ Reply

Links

1. https://www.learncpp.com/author/Alex/
2. https://ikeamenu.com/humix/video/Ipms8sWsckf
3. https://en.wikipedia.org/wiki/Hello_world
4. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#creating_project_vs19
5. https://www.learncpp.com/cpp-tutorial/installing-an-integrated-development-environment-ide/
6. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#visual_studio_solution_explorer
7. https://www.learncpp.com/cpp-tutorial/function-return-values-value-returning-functions/
8. https://www.learncpp.com/cpp-tutorial/a-few-common-cpp-problems/

9. https://www.learncpp.com/cpp-tutorial/introduction-to-the-compiler-linker-and-libraries/
10. https://www.learncpp.com/
11. https://www.learncpp.com/compiling-your-first-program/
12. https://www.learncpp.com/cpp-tutorial/statements-and-the-structure-of-a-program/
13. https://gravatar.com/
14. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-595423
15. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-594574
16. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-594480
17. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-594557
18. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-593987
19. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-593705
20. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-593499
21. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-593240
22. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-593091
23. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-592785
24. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-592876
25. https://www.learncpp.com/cpp-tutorial/compiling-your-first-program/#comment-591910